NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ALLOCATION OF UAV SEARCH EFFORTS USING DYNAMIC PROGRAMMING AND BAYESIAN UPDATING

by

Kevin K. McCadden
Christopher A. Nigus

June 2008

Thesis Advisor: Johannes O. Royset
Second Reader: Moshe Kress

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Allocation of UAV Search Efforts using Dynamic Programming and Bayesian Updating		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Kevin McCadden and Christopher Nigus			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

As unmanned aerial vehicle (UAV) technology and availability improves, it becomes increasingly more important to operate UAVs efficiently. Utilizing one UAV at a time is a relatively simple task, but when multiple UAVs need to be coordinated, optimal search plans can be difficult to create in a timely manner. In this thesis, we create a decision aid that generates efficient routes for multiple UAVs using dynamic programming and a limited-look-ahead heuristic. The goal is to give the user the best knowledge of the locations of an arbitrary number of targets operating on a specified graph of nodes and arcs. The decision aid incorporates information about detections and nondetections and determines the probabilities of target locations using Bayesian updating. Target movement is modeled by a Markov process. The decision aid has been tested in two multi-hour field experiments involving actual UAVs and moving targets on the ground.

14. SUBJECT TERMS Unmanned Aerial Vehicle, Search Model, Dynamic Programming, Bayesian Updating, Simulation, Multiple Searcher Routing.			**15. NUMBER OF PAGES** 87
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ALLOCATION OF UAV SEARCH EFFORTS USING DYNAMIC PROGRAMMING AND BAYESIAN UPDATING

Kevin K. McCadden
Ensign, United States Navy
B.S., United States Naval Academy, 2007

Christopher A. Nigus
Ensign, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN APPLIED SCIENCE
(OPERATIONS RESEARCH)**

from the

**NAVAL POSTGRADUATE SCHOOL
June 2008**

Authors: Kevin K. McCadden

Christopher A. Nigus

Approved by: Johannes O. Royset
Thesis Advisor

Moshe Kress
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As unmanned aerial vehicle (UAV) technology and availability improves, it becomes increasingly more important to operate UAVs efficiently. Utilizing one UAV at a time is a relatively simple task, but when multiple UAVs need to be coordinated, optimal search plans can be difficult to create in a timely manner. In this thesis, we create a decision aid that generates efficient routes for multiple UAVs using dynamic programming and a limited-look-ahead heuristic. The goal is to give the user the best knowledge of the locations of an arbitrary number of targets operating on a specified graph of nodes and arcs. The decision aid incorporates information about detections and nondetections and determines the probabilities of target locations using Bayesian updating. Target movement is modeled by a Markov process. The decision aid has been tested in two multi-hour field experiments involving actual UAVs and moving targets on the ground.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Since its conception, the unmanned aerial vehicle (UAV) has been a coveted battlefield asset. The ability of these vehicles to perform reconnaissance and attack missions while keeping the operators directly out of harm's way creates an advantage in the domains of information gathering and force protection. UAVs have only recently been introduced on the battlefield in significant numbers, and the ability to operate multiple UAVs efficiently and effectively can be improved further.

This thesis creates a decision aid that provides efficient search routes for multiple UAVs searching for multiple targets operating on a known graph of nodes and arcs. The decision aid dynamically provides estimates of target locations during its use.

The decision aid consists of a dynamic program that is solved approximately using a two-timestep look-ahead heuristic. Target location probabilities are computed using Bayesian updating based on the detections and nondetections from the previous timestep. The decision aid includes the possibility for UAVs to go on and offline due to mechanical difficulties or limited endurance.

The decision aid was tested in two field experiments at Camp Roberts, California, as part of the USSOCOM-NPS Field Experimentation Program. The field experiments included up to three UAVs and five target vehicles. For the second experiment, a prototype of the decision aid running through a Microsoft Excel user-interface was used. The interface proved to be highly effective in communicating to the user

the current knowledge of target locations and provided timely recommendations for the UAV operators.

ACKNOWLEDGMENTS

We would like to especially thank professors Royset, Kress, and Chung for all of their help in the design and experimentation of our model. We would also like to thank Anton Rowe for his work on the Excel interface. Completing this thesis is an important step in furthering our education and a milestone in our young careers and we could not have done it without all of your help and guidance. We would also like to thank all of our other professors from NPS as well as our family and friends who helped us through this difficult process.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

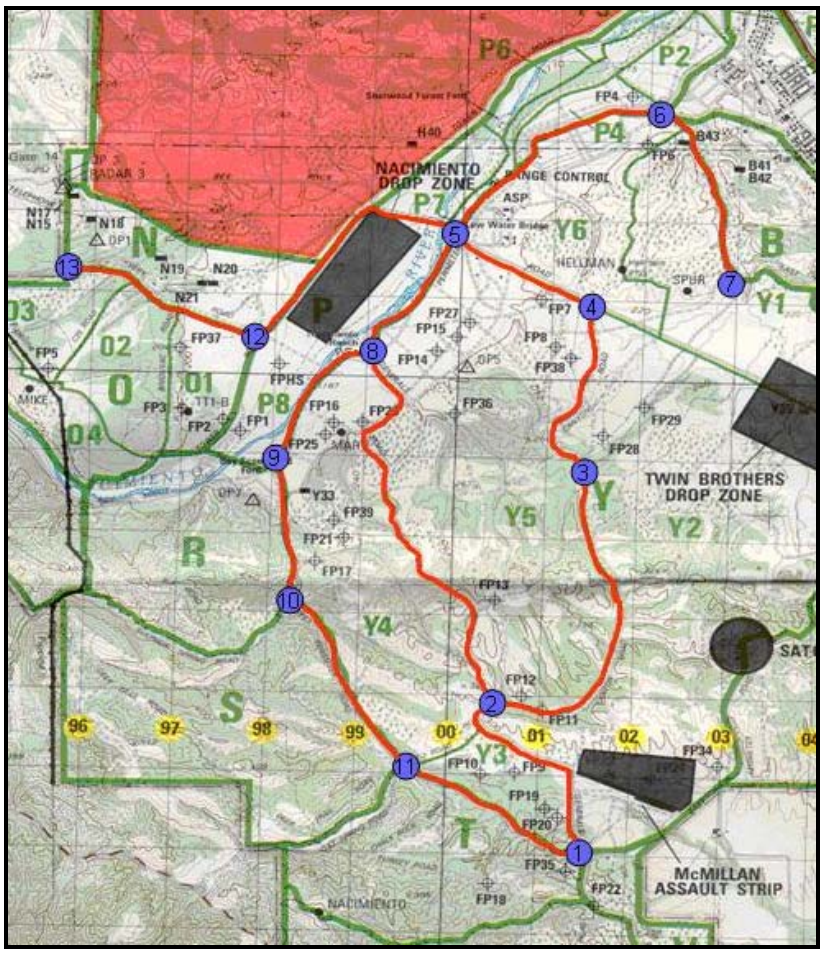
A. MOTIVATION AND PROBLEM DEFINITION

Search for moving targets arises in many different contexts. For example, searching is necessary when the goal is to find drug smugglers or shot-down pilots during search and rescue missions. The sensors used for these searches are often mounted on unmanned aerial vehicles (UAVs), thus UAVs become search assets. When multiple UAVs interact during a search, there becomes a need to effectively operate and manage them within the search environment.

We consider a finite number of searchers and targets that move on a graph of nodes and arcs. We assume the searchers have a close estimate of the number of targets. The targets remain within the graph and move according to a known Markov process. The overall goal is to route the searchers during a finite time horizon so that the search coordinator gains the maximum situational awareness of all targets, as quantified by probability distributions of target locations. There are many possible objective functions for problems of this kind. We specifically aim to maximize the expected number of detected targets until the finite time horizon while ignoring targets that are known to be located at a given location with a probability larger than a specified threshold. Target thresholds are discussed in detail in section A of Chapter II. We refer to this problem as the search optimization problem (SOP). In this thesis, we develop a model for SOP and a heuristic algorithm for obtaining efficient search plans in real-time within a rolling time horizon framework.

The graph in SOP could represent a road network where nodes are intersections and arcs are roads. Alternatively, the graph could represent a grid of area cells on the open ocean. Figure 1 shows an example of nodes and arcs in a road network at Camp Roberts, California.

Figure 1. Example of Graph.



Currently, no tractable model of SOP exists that incorporates all major aspects of real-world operations. SOP is difficult to solve optimally because the optimal move for the searchers at a timestep is dependent on the future searcher locations and actions as well as target location probabilities. We refer to such locations, actions, and

probabilities at a particular point in time as the “state” at that time. This dependence on future states requires the use of dynamic programming. This situation tends to result in intractable model formulations of SOP that cannot be solved quickly enough for use in a real-time decision aid. Dynamic programming is discussed in subsection B2.

In this thesis, we develop a new version of a decision aid called Aerial Search Optimization Model (ASOM), see, e.g., [12]. It consists of a tractable model for SOP, an associated heuristic algorithm for generating search policies, and a user interface. ASOM is specifically tailored for use by UAV operators, provides effective UAV routes quickly, and is relevant to many different search applications.

B. FUNDAMENTAL CONCEPTS

1. Bayesian Updating

Bayesian updating in the context of search is a process that begins with prior knowledge of target location probabilities, commonly referred to as the *a priori* map. This map is based on previous information, if such info exists, or it is assumed to be uniform, absent prior information. Figure 2 gives an example of a 4 cell *a priori* map where a single searcher is searching for a single stationary target known to be present in the map. In this thesis, we account for false negatives, but we assume that a searcher will not report a target on a node or an arc if there is no target at that node or arc (i.e., no false positives). We refer to Chung and Burdick [3] for a discussion of false positive reports. If the searcher looks

in the top left cell and fails to find the target, then Figure 3 shows the resulting posterior map given the searcher has a .5 conditional probability of detection. The posterior map is computed by the following equation:

$$P(A_i \mid D') = \frac{P(D' \mid A_i)P(A_i)}{\sum_j P(D' \mid A_j)P(A_j)} \qquad \forall i$$

where

i, j	index of target cells
$P(A_i)$	probability target is located in area i
$P(D' \mid A_i)$	probability of no detection in cell i given target is in cell i
$P(A_i \mid D')$	probability target is located in cell i given no detection is made in that cell

For each cell, the updated probabilities are computed by multiplying the probability of no detection given there is a target in the cell by the prior probability there is a target in the cell. This number must then be divided by the sum of these numbers for all cells in order to normalize the probabilities. See Wagner, Mylander, and Sanders [20] for a more detailed mathematical explanation of Bayesian Updating.

Figure 2. A Priori Target Distribution.

.40	.30
.20	.10

Figure 3. Posterior Target Distribution.

.25	.375
.25	.125

The above discussion deals with “false negatives,” which occur when a searcher fails to detect a target that is actually there.

2. Dynamic Programming

Dynamic programming is a framework for modeling decisions made over time [14]. The state of a dynamic program is a snapshot of the system being modeled at a specific time. Given a finite time horizon, the backward recursion algorithm generates optimal decisions at every timestep starting from the end and working backwards assuming there are a finite amount of states. However, this involves examining all states at each time step and determining the best decision at that state.

The backward recursion algorithm breaks down if there are an infinite number of states and/or the determination of the best decision at a state is a difficult optimization problem. In addition, it may be problematic to use this algorithm if the time horizon is not known.

Approximate dynamic programming algorithms seek to overcome the shortcomings of the backward recursion algorithm by introducing approximations. There exist a large number of approximate dynamic programming algorithms, see, e.g., [14]. Typically, these algorithms step forward

in time. The main difficulty is to determine the “value” of transitioning to a specific state. One technique is to use a limited look-ahead. This is a process of enumerating all possible moves for all timesteps of the designated look-ahead period and making the moves that achieve the greatest reward in terms of the objective function. Longer look-ahead periods will better approximate the optimal dynamic programming solution. We will use an approximate dynamic programming algorithm because it provides an effective solution that can be provided in real-time, a key requirement for our implementation.

C. PAST WORKS

The goal of the constrained-path, moving-target search problem [5, 6, 7, 13, 18, 19, 21] is to find the search route that maximizes the probability of target detection within a fixed time. The classic setup involves a single searcher and a single target moving within a finite number of cells in discrete time. Both the searcher and the target are allowed to occupy a single cell each timestep, and detections may only occur when the searcher and target occupy the same cell. Detection probabilities can be based on sensor data or derived from the random search formula [22]. The target’s probability distribution is maintained through Bayesian updates for nondetection each timestep if the target is not found.

For the classic constrained-path, moving-target search problem, Eagle and Yee [6] select a searcher route over a given number of time periods that minimizes the probability of nondetection. Their formulation is a non-linear program with linear constraints, which allows one to apply

Zangwill's [24] Convex Simplex Method (CSM). Eagle and Yee [6] create a myopic search, and while results of their example show the CSM solution to always be optimal, the myopic search may not provide a good approximation of the optimal solution.

A partially observable Markov decision process [2] is another concept that has been applied to the constrained-path, moving-target search problem. The idea is that a decision must be made based on partial information, and the outcome of the decision is unknown until after it has been made. The search application is well-suited for this setup because the searcher will have incomplete knowledge of target location after each timestep based on the updated target probability distribution. The searcher will not know whether or not the search will be successful until after the new search route is chosen.

Eagle [5] provides an optimal solution technique using dynamic programming and assuming a finite time horizon. He uses a partially observable Markov decision process, which is faster than standard linear programming methods because total enumeration is limited to searching only the cells one can reach from the searcher's previous location. Stewart [18, 19] creates an approximate solution procedure using branch-and-bound techniques. Eagle and Yee [7] extend Stewart's work and create a branch-and-bound method that produces optimal solutions and is faster than the dynamic programming approach. Washburn [21] creates a branch-and-bound approach as well. Both Eagle and Yee [7] and Washburn [21] consider searchers that have continuous search routes. Other than Washburn [21], who accounted for

multiple searchers, these problems consider one searcher against a single target and provide optimal solutions.

Dell, Eagle, Martins, and Santos [4] extend the problem to include multiple searchers. They create a branch-and-bound procedure to optimally solve the problem as well as six heuristics that take four different approaches to the problem: solve partial problems optimally, maximize the expected number of detections, implement a genetic algorithm, and use local searches with random restarts. The partial problem technique involves a moving horizon where each one is solved optimally using branch-and-bound.

Members of the autonomous systems and control community have analyzed the multiple UAV search problem as well. Some utilize recursive Bayesian filtering [1, 10] while others focus on cooperative control [8, 11] and decentralized search [1] techniques. Many of them have considered the problem of multiple searchers looking for multiple targets [1, 8, 9, 10, 11, 23], which is an extension to the works mentioned above [5, 6, 7, 13, 18, 19, 21]. Fernandez, Flint, and Polycarpou [9] as well as Chung and Burdick [3] create a Bayesian method that helps take into account false positives.

Another consideration is using discrete time to more closely model continuous time. This situation occurs when the travel time for targets and searchers between cells is not a multiple of the discrete timestep. Lau, Huang, and Dissanayake [13] enhance the branch-and-bound method to take into consideration the non-uniformity of the search environments. They develop a new bound that leads to faster solution times as well as the possibility of better solutions when the environment being modeled is spatial-

temporal non-uniform in nature. Sato and Royset [17] produce alternative bounds and even faster solutions.

In the near future, sufficient technology will exist to allow the automatic detection of targets by computer systems. When these automatic detections can be incorporated within a search program, it will allow the autonomous routing of UAVs. With current technology, human operators are required to visually identify targets. The issue of target detection can be handled with a decision aid that has an input for the detections made each timestep.

While many solutions have been presented for the constrained-path moving-target search problem and some research tools have been developed for specific scenarios (see, e.g., [15, 16]), a decision aid that can be used in real-world scenarios has yet to be fully developed. The goal of our research is to provide a user-friendly decision aid that is capable of creating efficient UAV routes for detecting multiple targets operating on a known graph. This decision aid will be capable of providing real-time effective decisions with computation times on the order of seconds.

D. STRUCTURE OF THESIS AND CHAPTER OUTLINE

This thesis is divided into five chapters, including the Introduction. Chapter II discusses the development of the model and the dynamic programming formulation. Chapter III introduces the actual algorithm used to implement our model. Next, it analyzes the accuracy and runtime of our heuristic approach. Finally, it discusses the Excel user-interface created for our decision aid. Chapter IV talks about our field experiments in Camp Roberts, California and

explains some of the updates our decision aid underwent in the process. Chapter V gives several conclusions from our work as well as recommendations for future work involving ASOM.

II. MODEL

A. MODEL DEVELOPMENT

We formulate a model of SOP using dynamic programming with Bayesian updating. We assume that each target moves according to a Markov process and that the targets move independently of one another. The presentation below and our implementation of the model assume that all the Markov processes for the various targets have the same transition matrices. However, it is trivial to extend this to the general case where targets follow different movement processes. Targets are differentiated by their velocity and type characteristics (e.g., person versus vehicle).

The searchers are differentiated by a variety of characteristics including name, velocity, sweepwidth of their sensors, and whether or not they have a camera with a moving eye which enables them to search nearby roads while flying straight routes between nodes.

All dynamic programming models must have discrete timesteps. In our model, timesteps are used as a discrete representation of continuous time. One timestep is the length of time between each discretized value of time with smaller timesteps being a better approximation of continuous time.

Our dynamic programming model contains several states that change according to some process as the model advances through time by the use of timesteps. The state of the searcher includes the arc the searcher is currently on, the amount of time until the searcher reaches the head node of that arc, and the type of move that is currently being

executed. There are three possible types of moves: “Road Search,” “Transit,” and “Search at Location.” “Road Search” means that the searcher examines the road corresponding to the current arc while traversing it. It is possible to detect targets on that road, and any time remaining of the timestep after reaching the head node of the arc is spent searching that head node. “Transit” means that the searcher flies a direct route from the tail node to the head node. It is not possible to detect a target when completing this type of move, but rather offers the possibility to reach the head node faster and allows more time for search at that node. “Search at location” means that the searcher spends the entire timestep searching its current location.

The other main states in the dynamic programming model are the target probability maps. There is one probability map for each target and the entire map is a matrix where the entry in row i and column j represents the probability that the target is on arc (i,j), if $i = j$, this represents the probability at a node. These probability maps are dynamically updated as the model transitions from one timestep to another. The updates due to detections and nondetections using Bayesian updating are first carried out. Then, the updates due to movement of targets by the Markov process are computed.

More specifically, when detections are made, the location and type of detection are inputted into the model. The model updates the target probability maps for the detections based on the probabilities of seeing different targets at the input detection locations. It looks at all the different “detection scenarios” and determines the probability of each happening and decides which scenario

occurred based on a random draw with the associated probabilities. Here, a “detection scenario” is an element of the set of all the different permutations of possible target detections at each detection location. For example, if there are two detections at time t and three available targets, the model creates all possible permutations of target detection scenarios. In this situation, there are six possible scenarios, three choices (possible targets) for the first detection and then two remaining choices (one of the two not found in the first detection) for the second detection. The model then computes the probability of each of the six different scenarios occurring based on the target marginal probabilities and decides which one actually occurred using a random draw with the corresponding probabilities.

We also use the concept of search thresholds. This threshold is a user input between 0 and 1 used to determine what level of target knowledge will constitute “knowing” where a target is located. This is an attempt to gain better total situational awareness by ignoring targets that we “know” are at certain locations. A threshold value of 1 creates a greedy policy where searchers will circle targets unless a higher probability mass presents itself at a nearby location. On the contrary, if the threshold value is less than 1, then targets whose maximum probability mass is above that threshold will not be searched for, resulting in a less greedy policy.

We also calculate an aggregate probability map to represent the normalized probability of all targets that are unknown (i.e., do not reach the threshold) by summing the

probability mass of all unknown targets at each location and dividing it by the number of these targets.

SOP is defined in terms of some finite time horizon. This may be related to the endurance of the searchers (e.g., UAV flight time) or operational considerations. In practice, the time horizon may not be completely known. Looking further into the future with a dynamic program will give better decisions in the current timestep than a shorter look-ahead. To limit computing time and allow for a real-time decision aid, we only consider a two time-step look ahead, i.e., we set the time horizon in SOP to two. We call this the two timestep look-ahead problem (TTLP). The objective function in TTLP, which we maximize, is the expected number of target detections at all arcs and nodes visited during a given sequence of two moves for all searchers. In determining the aggregate probability mass for the second time period, the objective function assumes that there are no detections during the first timestep. The TTLP can be solved optimally by total enumeration, but as the number of searchers increases, the computational effort increases exponentially. As a result, we constructed a heuristic algorithm for solving TTLP. The heuristic algorithm amounts to total enumeration of all solutions of a simplified two timestep look-ahead problem (STTLP) which we describe next. The mathematical formulation of STTLP follows in Section B.

STTLP is identical to the TTLP except that it involves a simplified objective function. The STTLP objective function, as in TTLP, is the expected number of detections, but now the expected number of detections is computed slightly differently in the second timestep. The

probability mass present in the second timestep is calculated for each searcher independently (with no conflictions of moves), only taking into account probability updates for that particular searcher's previous move (not all previous moves as in the TTLP). As with the TTLP, it is assumed that there are no detections during the first timestep. All states and arrays that are relevant to this update are labeled with the superscript "ND" (no detection).

The following is an example of the flow of ASOM. After the initial states are established, the searchers are given starting locations. If there are no initial detections, ASOM recommends searcher moves based on the STTLP. For each timestep, detections are entered and ASOM reoptimizes the recommended searcher moves for the next timestep given there are no more detections. At this point, the operator can either accept the recommendations or enter in alternate searcher moves. This process is repeated for each timestep until the search is completed.

B. DYNAMIC PROGRAMMING FORMULATION OF STTLP

For notational convenience, we use $\bullet$ to denote the use of an array of all the available values for that index, thus for some values $X_{i,j}$, then $X_{\bullet,j} = \left(X_{1,j}, X_{2,j}, ..., X_{|I|,j}\right)^T$.

Indices

i, j, k	Nodes
m	Searcher
t	Timestep
u	Target
b	Types of targets

Sets

M	Set of all available searchers, $m \in M$.
I	Set of all available nodes, $i, j, k \in I$.
T	Set of timesteps, $t \in T$.
U	Set of targets, $u \in U$.
B	Set of target types, $b \in B$.
$R \subset I \times I$	Subset of pairs of nodes (*i, j*) representing arcs for which there is a road connecting *i* to *j*, $(i, j) \in R$. Also, $(i, i) \in R$, $\forall i \in I$.
$Q \subset I \times I$	Subset of pairs of nodes (*i, j*) representing possible transit arcs between *i* and *j*, $i, j \in I$.

Data

$DISTANCE_{i,j}$	Distance along road corresponding to arc (*i,j*) (mi), $(i, j) \in R$.
$TRANSIT_{i,j}$	Straight-line distance between nodes *i* and *j* (mi), $(i, j) \in Q$.
$SEARCHARC_m$	1 if searcher *m* searches a road while on transit arcs, 0 otherwise, $m \in M$.
$SPEED_m$	Constant speed of searcher *m* (mph), $m \in M$.
SW_m	Sweep width of searcher *m* (mi), $m \in M$.
$SPEEDT_u$	Speed of target *u*, $u \in U$.
STEP	Duration of timestep (minutes).
$START_m$	Starting node of searcher *m*, $m \in M$.

$PD_{i,j,m}$ Probability of detecting a target on the road corresponding to (i, j) for searcher m given that a target is on the road, $(i, j) \in R$, $m \in M$. If $i = j$, then $PD_{i,j,m} = 0$ since detections at a node is determined by function $PDET_{i,m}(\tau)$, defined later.

$MATRIX_{i,j}$ Probability of a target moving onto arc from node i to node j, $i, j \in I$.

$TTS_{i,j,u}$ Target timesteps calculation, the amount of timesteps target u takes to travel arc (i, j), $(= 60 DISTANCE_{i,j} / ((STEP)(SPEEDT_u)))$, $(i, j) \in R$, $u \in U$.

$THRESHOLD$ An input threshold between 0 and 1 to determine what level of target knowledge will constitute "knowing" where a target is.

$TURN$ Constant probability that a target travelling along an arc (i, j), $(i, j) \in R$ will turn around and go the other way.

$TYPE_u$ The type of target u, $u \in U$, $TYPE_u \in B$.

The following decision variables are computed at every time $t \in T$.

Decision Variables at Timestep *t*

$x_{i,j,m,t}$ 1 if searcher m is traveling from i to j, 0 otherwise.

$y_{m,t}$ Time until searcher m completes the recommended move (hrs).

$z_{m,t}$ 1 if searcher *m* is searching, 0 otherwise.

$V_{m,t} = (x_{\bullet,\bullet,m,t}, y_{m,t}, z_{m,t})^T$

Variable Array for searcher *m*.

$X_t = (x_{\bullet,\bullet,\bullet t}, y_{\bullet,t}, z_{\bullet,t})^T \quad \left(= V_{\bullet,t}\right)$

Variable Array for all searchers.

States at time *t*

$SEARCHER_{m,t} = (i, z_{m,t-1}, y_{m,t-1})^T \quad \forall m$, where

i Current Location/Destination;

$z_{m,t-1}$ 1 if searching, 0 if transiting from previous timestep (Assume 1 if $t = 1$);

$y_{m,t-1}$ Time to completion of the move from the previous timestep for searcher *m.* (hrs) (Assume 0 if $t = 1$).

$MARG_{i,j,u,t}$

Probability of target *u* being on arc (*i*, *j*). $(i, j) \in R$, $u \in U$, $t \in T$.

$$AGG_{i,j,t} = \frac{\sum_{u \in U | \max(MARG_{\bullet,\bullet,u,t}) < THRESHOLD} MARG_{i,j,u,t}}{\sum_{u \in U | \max(MARG_{\bullet,\bullet,u,t}) < THRESHOLD} 1} \quad \forall i, j$$

Aggregate probability of all targets being on arc (*i*, *j*), $(i, j) \in R$, $t \in T$.

$S_t = (SEARCHER_{\bullet,t}, MARG_{\bullet,\bullet,\bullet,t})^T$

State Vector.

$MARG^{ND}_{i,j,u,m,t}$

Probability of target u being on arc (i, j) according to the viewpoint of searcher m. $(i, j) \in R$, $u \in U$, $m \in M$, $t \in T \setminus \{1\}$.

$$AGG^{ND}_{i,j,m,t} = \frac{\sum_{u \in U | \max(MARG^{ND}_{\bullet,\bullet,u,m,t}) < THRESHOLD} MARG^{ND}_{i,j,u,m,t}}{\sum_{u \in U | \max(MARG^{ND}_{\bullet,\bullet,u,m,t}) < THRESHOLD} 1} \qquad \forall i, j, m$$

Aggregate probability of all unknown targets being on arc (i, j), $(i, j) \in R$ from the viewpoint of searcher m, $m \in M$, $t \in T \setminus \{1\}$.

$S^{ND}_{m,t} = (SEARCHER_{\bullet,t}, MARG_{\bullet,\bullet,\bullet,t})^T$

The current state according to searcher m. This is only used in the future look-ahead, $t \in T \setminus \{1\}$.

In the next two sections on functions and random inputs, parts of the formulation are not included for notational convenience. For a complete list, see Appendix I.

Random variables and sets during time *t*

$D_{i,j,b,t}$ Number of detections of type b on arc (i, j) during time t, $(i, j) \in R$, $t \in T$.

Functions

$R_t(S_t, X_t)$

Reward for all searchers traveling between node i and j, $m \in M$, $i, j \in I$.

$R^{ND}_{m,t}(S^{ND}_{m,t}, V_{m,t})$

Reward for searcher m traveling between node i and j, $m \in M$, $i, j \in I$. This function is only used in calculating the future reward when there is only knowledge of the searcher m.

$PDET_{i,m}(\tau)$

Probability of detection at node i by searcher m, dependent on amount of time searched, τ, $i \in I$, $m \in M$.

$NEGATIVE_{i,j,u,t}(S_t, X_t)$

Function to update probability maps for failed detection via Bayesian updating.

$NEGATIVE^{ND}_{i,j,u,m,t}(S_t, V_{m,t})$

Function to update probability maps for failed detection via Bayesian updating for look-ahead. Heuristic approach only takes into account the move of searcher m.

$MARKOV_{i,j,u,t}(S_t)$

Function to update probability maps for target movement based on Markov matrix.

$MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t})$

Function to update probability maps for only the movement of target m. It is used in the calculation of the "no detection" marginals according to searcher m.

$POSITIVE_{i,j,u,t}(S_t, D_{\bullet,\bullet,\bullet,t})$

Function to update probability maps for positive detection via Bayesian updating.

Policy: Set $X_t = X_t^*$, where (X_t^*, X_{t+1}^*) is the optimal solution of the simplified two timestep look-ahead problem (STTLP):

$$\max_{X_t, X_{t+1}} \quad R_t(S_t, X_t) + \sum_m R_{m,t}^{ND}(S_{m,t+1}^{ND}, V_{m,t+1})$$

Subject to:

$$\sum_{m,i} \left(x_{i,j,m,t} + \sum_{m' \in M \setminus m} x_{i,j,m',t+1} \right) \leq 1 \qquad \forall j$$

(Do not allow overlapping of moves)

$$\sum_m x_{i,j,m,t} \leq 1 \qquad \forall i, j$$

(Max one searcher per arc at time t)

$$\sum_m x_{i,j,m,t+1} \leq 1 \qquad \forall i, j$$

(Max one searcher per arc at time t + 1)

$$\sum_{i,j} x_{i,j,m,t} \leq 1 \qquad \forall m$$

(One move per searcher at time t)

$$\sum_{i,j} x_{i,j,m,t+1} \leq 1 \qquad \forall m$$

(One move per searcher at time t + 1)

If searcher m is at node i at time t, then:

$$\sum_j x_{i,j,m,t} = 1 \qquad \forall m$$

(Must start at the starting position)

End if

$$\sum_{(i,j) \in R} x_{i,j,m,t} \geq z_{m,t} \qquad \forall m$$

(Tracks transiting/searching at time t)

$$\sum_{(i,j) \in R} x_{i,j,m,t+1} \geq z_{m,t+1} \qquad \forall m$$

(Tracks transiting/searching at time t + 1)

$\forall m$, If $y_{m,t} \le STEP$, then:

$$\sum_i x_{i,j,m,t} = \sum_k x_{j,k,m,t+1} \qquad \forall j$$

Else:

$$\sum_i x_{i,j,m,t} = x_{j,j,m,t+1} \qquad \forall j$$

End if

(Continuity of route)

If $t = 1$, then:

$$y_{m,t} = \frac{60}{STEP} \frac{\sum_{i,j} \left(x_{i,j,m,t} \left(DISTANCE_{i,j} z_{m,t} + TRANSIT_{i,j} \left(1 - z_{m,t} \right) \right) \right)}{SPEED_m} \qquad \forall m$$

Else If $t \ge 2$, then:

$$y_{m,t} = \max \left(y_{m,t-1} - STEP, \left(\frac{\sum_{i,j} \left(x_{i,j,m,t} \left(\begin{matrix} DISTANCE_{i,j} z_{m,t} + \\ TRANSIT_{i,j} \left(1 - z_{m,t} \right) \end{matrix} \right) \right)}{SPEED_m STEP / 60} \right) \right) \qquad \forall m$$

(Keeps track of timesteps until searcher m is available)

End If

$x_{i,j,m,t} \in \{0,1\} \qquad \forall i, j, m$

$x_{i,j,m,t+1} \in \{0,1\} \qquad \forall i, j, m$

$z_{i,j,m,t} \in \{0,1\} \qquad \forall i, j, m$

$z_{i,j,m,t+1} \in \{0,1\} \qquad \forall i, j, m$

$y_{m,t} \ge 0 \qquad \forall m$

$y_{m,t+1} \ge 0 \qquad \forall m$

Dynamics (Given S_t and X_t)

$\forall j, m$, if $\sum_i x_{i,j,m,t} > 0$, then:

$$SEARCHER_{m,t+1} = (j, z_{m,t}, y_{m,t})^T$$

Sets the searcher's state to the decisions of that searcher for this timestep.

End If

$$MARG_{i,j,u,t+1} = MARKOV_{i,j,u,t}(NEGATIVE_{\bullet,j,u,t}(POSITIVE_{\bullet,\bullet,u,t}(S_t, D_{\bullet,\bullet,\bullet,t}), X_t)) \quad \forall i, j, u$$

Updates the target marginals for the positive detection updates, the negative detection updates, and the movement of the targets based on the Markov process.

$$MARG^{ND}_{i,j,u,m,t+1} = MARKOV^{ND}_{i,j,u,m,t}(NEGATIVE^{ND}_{\bullet,j,u,m,t}(POSITIVE_{\bullet,\bullet,u,t}(S_t, ZEROS_{\bullet,\bullet,\bullet}), V_{m,t})) \quad \forall i, j, u, m$$

ZEROS denotes a matrix of zeros as input for the detection matrix, or "no detections found" in human input terms. The update only has knowledge of one searcher at a time, thus it calculates marginals

$$S_{t+1} = (SEARCHER_{\bullet,t+1}, MARG_{\bullet,\bullet,\bullet,t+1})^T$$

$$S^{ND}_{m,t+1} = (SEARCHER_{m,t+1}, MARG^{ND}_{\bullet,\bullet,\bullet,m,t+1})^T$$

Sets the regular and no detections state variable arrays.

THIS PAGE INTENTIONALLY LEFT BLANK

III. IMPLEMENTATION

A. MODEL IMPLEMENTATION

We implement the model in MATLAB version 7.0.1 and carry out all computational tests on a NES computer with a 1.83 gigahertz AMD Athlon XP processor and 512 megabytes of RAM. As described earlier, we implemented a heuristic solution to the TTLP, called STTLP. The code is written in many sub-functions so that a single aspect of ASOM can be changed without having to go through the entire code. The descriptions of our MATLAB functions are given in Appendix II.

B. HEURISTIC ACCURACY

The only straightforward method for ensuring that optimal searcher moves are chosen is total enumeration. The difficulty with total enumeration is that for every searcher added to the TTLP, the total number of searcher move combinations increases exponentially. Thus, we need the heuristic algorithm, STTLP (see section B in Chapter II). We compare our heuristic with the total enumeration approach in terms of runtime and accuracy to ensure it provides effective recommendations and that its speed improvements are worth sacrificing optimality. For one, two, and three searchers we create random target marginals, randomly place the searchers, and compare the moves recommended by our heuristic and total enumeration functions. We allow searchers to be “initially blocked” with a probability of .25. Here, “initially blocked,” means that the searchers

are constrained in their movements from the previous timestep (i.e., still in transit). This .25 probability represents the fact that during a normal run of our decision aid, the searchers make direct transits that require two timesteps and are blocked from making a new move for one timestep.

Table 1 shows the accuracy results of the heuristic for 1000 simulation runs. The accuracy is a ratio of the probability mass collected by the heuristic versus that collected by the total enumeration approach. It also displays the fraction of time the heuristic returns the optimal move. The "Within One Move of Optimal" column gives the fraction of time that the heuristic moves did not match up with the total enumeration moves for at most one searcher. Table 2 displays the runtimes of the heuristic and total enumeration approaches for one, two, and three searchers along with their 95% confidence intervals.

Table 1. Heuristic Accuracy Table.

Number of Searchers	Accuracy	Returns Optimal (TTLP) Move	Within One Move of Optimal (TTLP)
1	1	1	1
2	0.9914	0.944	0.985
3	0.9813	0.843	0.934

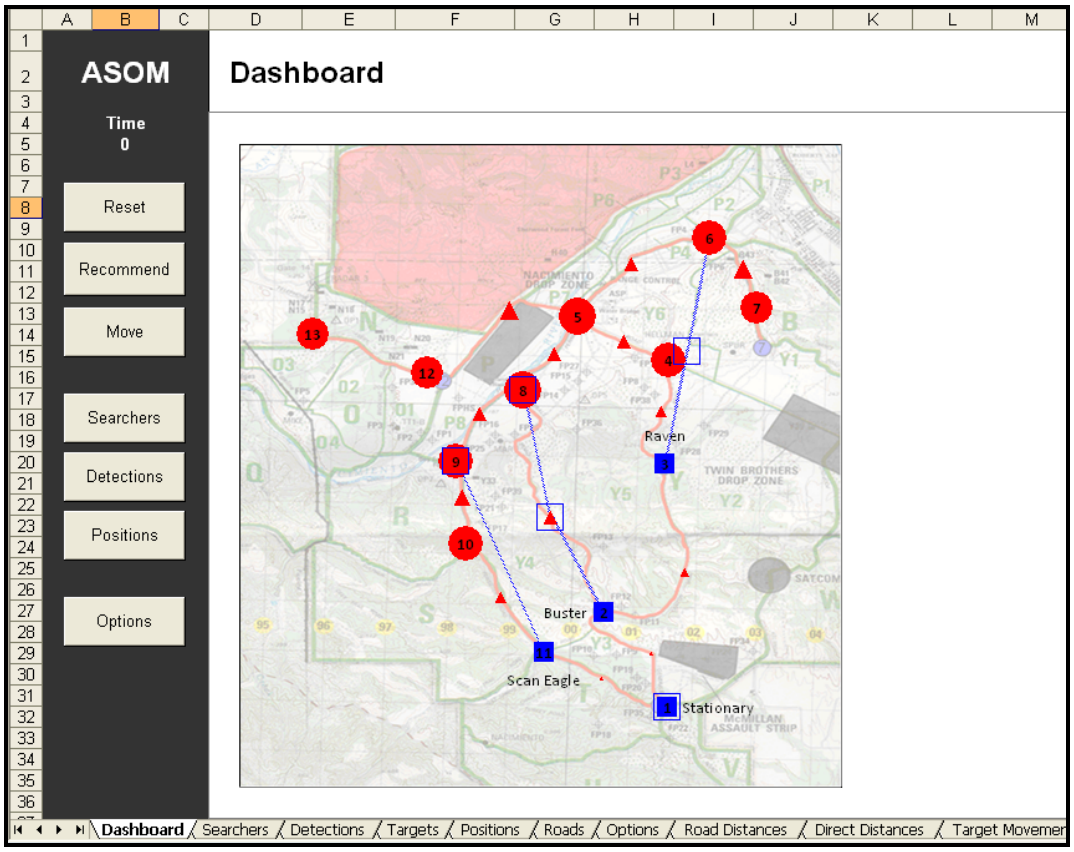
Table 2. Heuristic Runtime Table.

Number of Searchers	STTLP Runtime (sec)	Total Enumeration (TTLP) Runtime (sec)
1	.02165 +/- .00074	.01462 +/- .00074
2	.04219 +/- .00093	.8560 +/- .046
3	.07381 +/- .0076	64.21 +/- 4.45
4	.1046 +/- .00227	4186 (estimated)

C. EXCEL INTERFACE

The Microsoft Excel Interface was developed by Mr. Anton Rowe. Figure 4 is an example of the output display in the user interface.

Figure 4. Screenshot of Excel Interface.

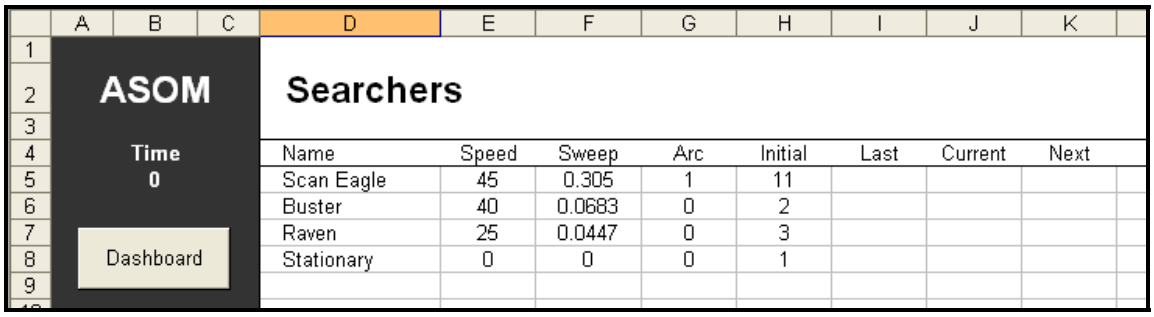


In Figure 4, the red circles represent all possible nodes and the red triangles represent all possible roads. The different sizes of the circles and triangles represent the aggregate probability of finding targets there. The solid blue boxes represent the different searchers at their current locations in this scenario. The blue lines and outlined boxes represent the recommended searcher moves for the current timestep. If a triangle is encased in the

outline of a blue box, this means the recommendation is to search the road to the corresponding node. A dotted blue line going straight to a node means transit directly to that node. If there is an outline of a blue box in the middle of a transit route, this means the searcher will not get to the designated node in one timestep and thus it is a directed move for the following timestep as well. If a searcher is stationary (zero speed) then the recommended move will always be to stay at the same location, shown by the blue outline around its current position. In the example above, Raven is transiting from node 3 to 6, but will take two timesteps to reach node 6. Buster is searching the road from node 2 to node 8 (one timestep) and Scan Eagle is transiting from node 11 to node 9 (one timestep).

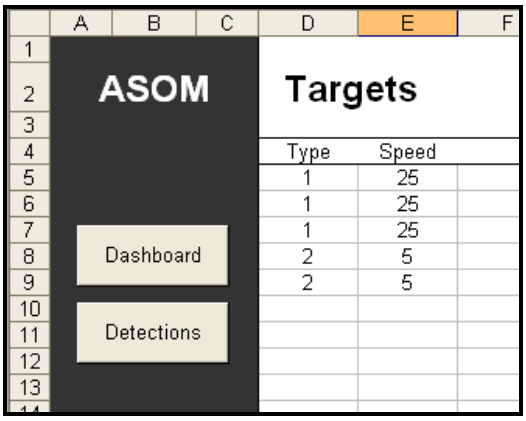
There are several required inputs for ASOM including parameters for both searchers and targets. For each available searcher, the name (as it will be displayed on the interface) should be provided, as well as the speed, sweepwidth, a binary entry for whether the UAV has a moveable camera capable of searching roads while flying straight line distances, and the starting position. An example input is seen in Figure 5. Notice there is also a stationary searcher in the scenario below, which is input by a searcher with speed equal to zero. A starting position must also be provided, but the “Sweep” and “Arc” categories for a stationary searcher are not used.

Figure 5. Example Searcher Input.



The available targets are simple inputs of the expected number and type of each target that will be available in the scenario. For each target, a speed and type must be provided, as seen in Figure 6. If the number of targets is not known, a reasonable estimate should be provided; the better the estimate the more accurate the model will be.

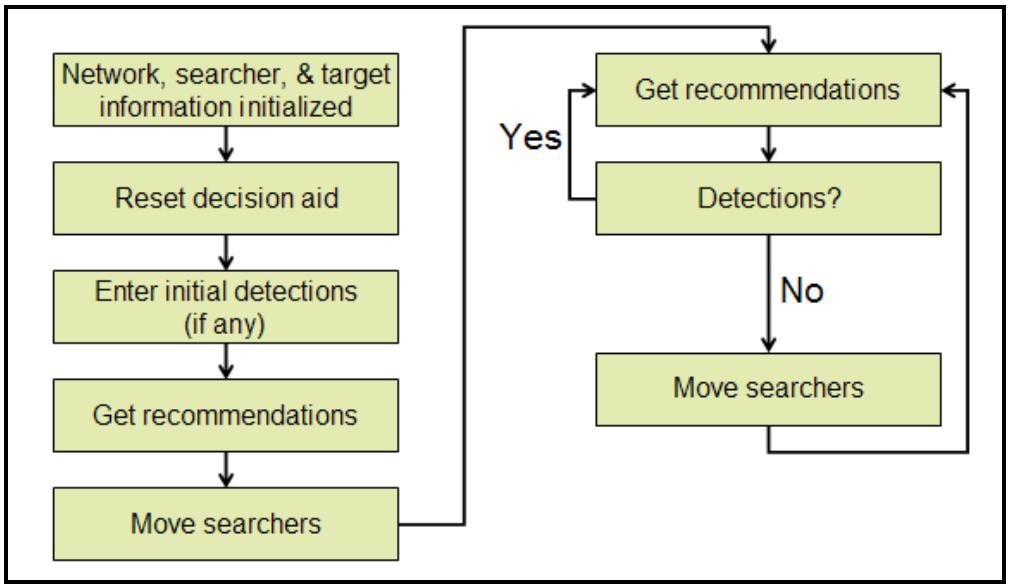
Figure 6. Example Target Input.



Detections are input during the current timestep of a model. The key feature here is the “Recommend” button. When pushed, this button gives recommendations based on the current state. If, however, detections are made between then and the end of the timestep, they can be inputted to update the state and a new set of moves will be outputted.

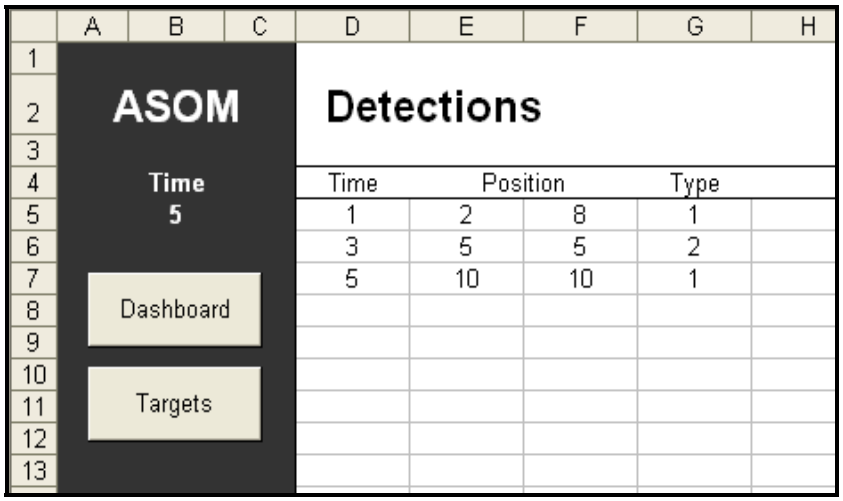
An example timeline of entering detections and moving targets can be seen in Figure 7.

Figure 7. User steps in ASOM.



Detections are inputted with four parameters: (i) timestep of the detection, (ii and iii) perceived starting node and ending node location of the target, and (iv) detection type. The starting and ending node location together represent the arc (i, j) (location) in which the target was detected, where if $i = j$, the target was detected stationary at node i; and if $i \neq j$, the target was detected on the road going from node i to node j. An example of what the target detection sheet might look like at timestep 5 can be seen in Figure 8. In this example, the first line says there was a detection of type 1 on the road from node 2 to node 8 at time 1. Similarly, the second line says there was a detection of type 2 stationary at node 5 at time 3.

Figure 8. Example Target Detections.



Additional data for ASOM include the latitude/longitude of the nodes, data for the roads (start/finishing nodes, length of the road, and latitude/longitude position to display the red triangle representing probability), direct distances between nodes (as a UAV can fly them), and the Markov movement matrix for each target.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. FIELD EXERCISES

We performed two field experiments in February and May 2008 at Camp Roberts, California using multiple Raven and Buster UAVs.

An important part of ASOM is the ability to take into consideration the needs of the operator and the possibility to react to unexpected situations. Several features of ASOM would not exist if we did not field test the decision aid and receive feedback from UAV operators. This allows ASOM to handle realistic scenarios in multiple environments.

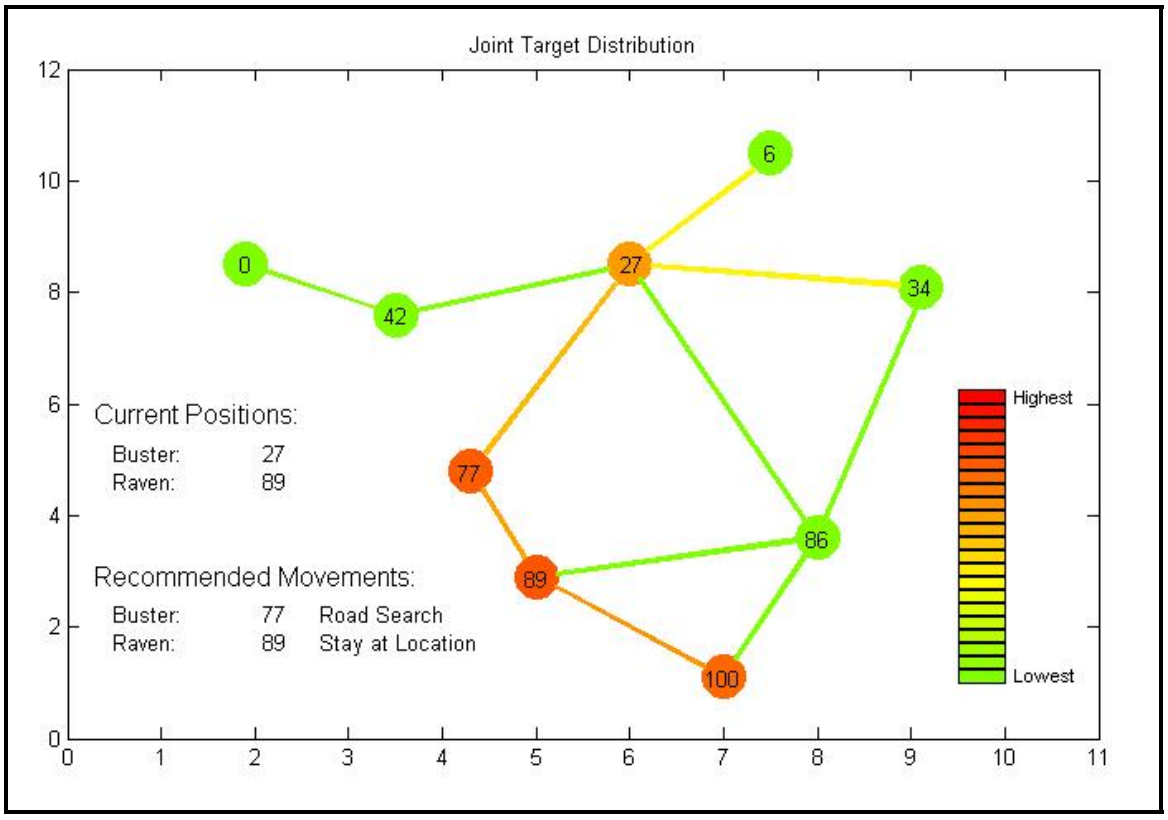
A. FEBRUARY EXPERIMENT

The purpose of the February experiment was to test a preliminary version of ASOM and make sure the results passed a reality check. A secondary purpose was to see what could be improved in the underlying code and what changes were necessary to make ASOM run smoother. There were several weather restrictions that limited the experiment, but overall the objective of the experiment was accomplished.

We ran our preliminary model with 5 moving targets (cars) traveling at 25 miles per hour and three searchers: one ground team, one Raven UAV, and one Buster UAV. ASOM isolated the possible location of the targets to one side of the map, as seen in Figure 9, and was correct in its judgment of possible target locations. In this preliminary version of the model, aggregate probability is given by a color scale rather than a size, with green representing the lowest probability, fading to yellow, then finally to red representing the highest probability. The nodes are still

represented by circles, but the roads are represented by straight lines between the nodes.

Figure 9. February Experiment Final Probability Map.



There were several important lessons learned from this experiment. The first stemmed from the fact that our approach was greedy in its search patterns. At this point, the searchers appeared to find a target and track it because this resulted in the largest reward while sacrificing knowledge of the other targets. This is not optimal if the objective is to maximize total knowledge of the system. We remedied this by creating the threshold input. As described earlier, this is equivalent to saying you “know” where a target is located if its maximum probability mass at any location is greater than the threshold probability.

Another change in ASOM was how to make the model more user-friendly than the current MATLAB code and input techniques. This was handled with a new Excel interface as discussed in the previous chapter. The usefulness of the interface is discussed in the May experiment section.

B. MAY EXPERIMENT

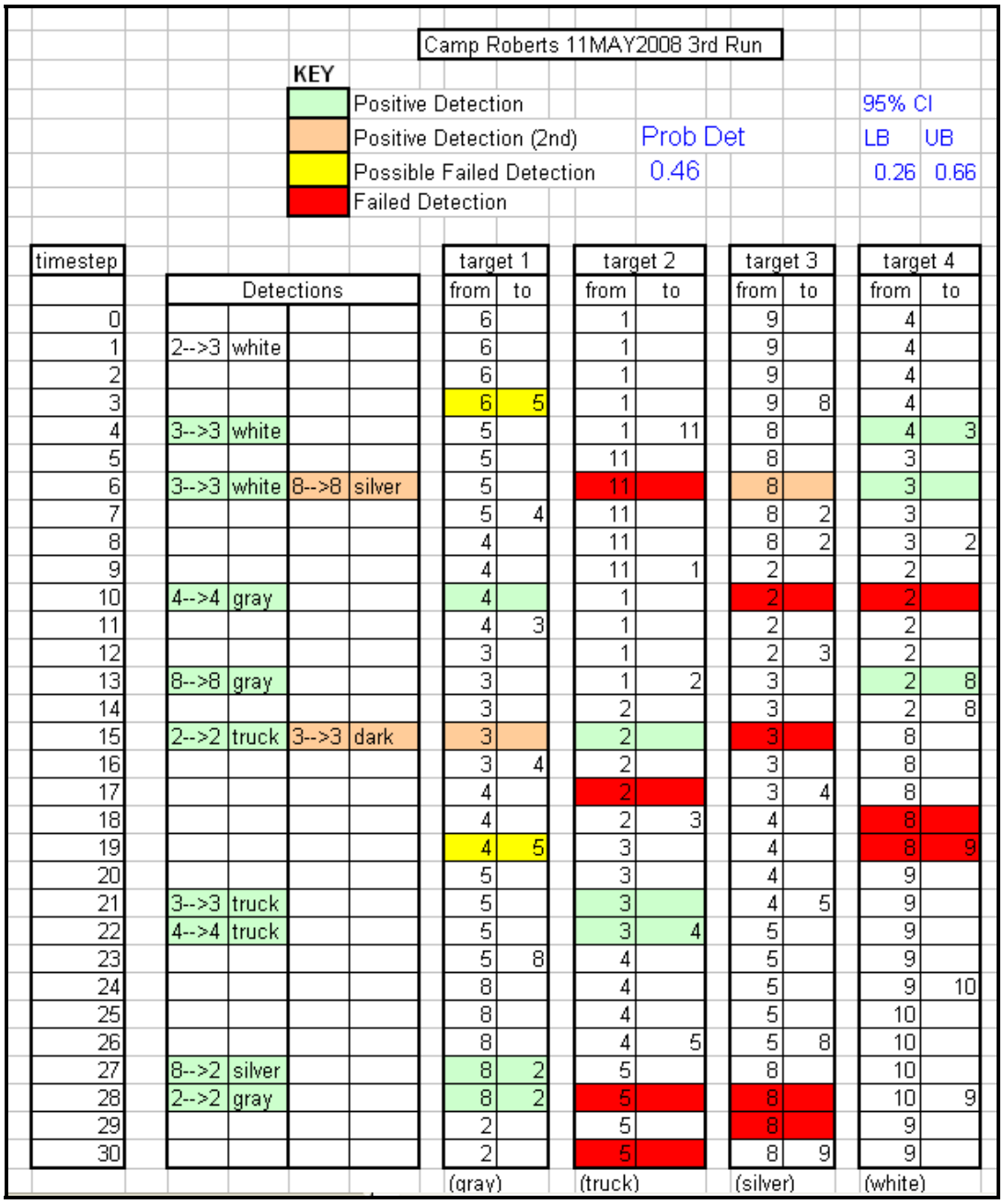
The goal of the May experiment was to test the updated code, which included the target threshold constraints to discourage a greedy policy which tracked detected targets. We implemented the Excel interface for the first time and evaluated its utility and functionality. The experiment was run with four targets (again, cars traveling at 25 miles per hour) and three searchers, one Buster UAV and two Raven UAVs.

The first day's trials led to the creation of the disabled node. This node is an abstract location where searchers are placed when they are refueling, damaged, or unusable. This allows ASOM to function in a larger set of scenarios as well as take into account unexpected events where a UAV becomes disabled. For example, in the first trial, the Buster UAV lost contact, deployed its parachute, and was unable to continue its search. The Raven UAVs also ran out of gas sooner than expected and had to land and refuel, thus cutting the experiment runs short.

The second day's trial utilized the disabled node update. This trial was extended to a nearly three hour scenario where UAVs were forced to refuel, thus testing the capabilities of the disabled node.

Figure 10 shows the locations of all of the targets and searchers as well as the color of the vehicles detected. The green and tangerine colored boxes represent actual target detections by the searchers. Yellow boxes represent possible failed detections, meaning the timing of the searcher or target leaving and the other arriving on location were close, but there could have been a failed detection. A red box means a target and a searcher were each at the same location, but there was no detection made at that time. From this, we calculated an estimate of the probability of detection with appropriate 95% confidence interval (0.46 +/- 0.20). Since the data set is relatively small, the confidence interval on the probability of detection is very wide. In any case, this might give us a better estimate on the actual probability of detection for these UAVs. In ASOM, the probability of detection is derived from the random search formula and is dependent on time as well as searcher characteristics, but it is generally higher than the above empirical estimate.

Figure 10. May Experiment Detection Results.



Camp Roberts 11MAY2008 3rd Run

KEY

- Positive Detection
- Positive Detection (2nd)
- Possible Failed Detection
- Failed Detection

Prob Det 0.46

95% CI LB 0.26 UB 0.66

timestep	Detections				target 1		target 2		target 3		target 4	
					from	to	from	to	from	to	from	to
0					6		1		9		4	
1	2-->3	white			6		1		9		4	
2					6		1		9		4	
3					6	5	1		9	8	4	
4	3-->3	white			5		1	11	8		4	3
5					5		11		8		3	
6	3-->3	white	8-->8	silver	5		11		8		3	
7					5	4	11		8	2	3	
8					4		11		8	2	3	2
9					4		11	1	2		2	
10	4-->4	gray			4		1		2		2	
11					4	3	1		2		2	
12					3		1		2	3	2	
13	8-->8	gray			3		1	2	3		2	8
14					3		2		3		2	8
15	2-->2	truck	3-->3	dark	3		2		3		8	
16					3	4	2		3		8	
17					4		2		3	4	8	
18					4		2	3	4		8	
19					4	5	3		4		8	9
20					5		3		4		9	
21	3-->3	truck			5		3		4	5	9	
22	4-->4	truck			5		3	4	5		9	
23					5	8	4		5		9	
24					8		4		5		9	10
25					8		4		5		10	
26					8		4	5	5	8	10	
27	8-->2	silver			8	2	5		8		10	
28	2-->2	gray			8	2	5		8		10	9
29					2		5		8		9	
30					2		5		8	9	9	
					(gray)		(truck)		(silver)		(white)	

Failed detections could stem from any combination of three sources of error. The searchers were at incorrect locations, the targets were at incorrect locations, or our estimation of the probability of detection for searchers finding targets was inaccurate. The problem of searchers

being at wrong locations seems unlikely because they are given GPS coordinates to fly to, and their locations are displayed on a screen. It is possible the targets (who were people driving around in cars) did not know the Camp Roberts map as well as we had hoped and were actually driving to wrong locations. The most likely source of error was that the camera feeds on the UAVs were scrambled enough that the operators had a hard time identifying targets, thus lowering our probability of detecting a target given a searcher and target were at the same location.

One other interesting aspect of having a long trial versus several short trials is a measurement of the situational awareness of the searchers. Specifically, the awareness of target location went in cycles. Examining Figures 11 and 12, the first is a picture showing UAV locations and target location probabilities half way through our second day's trial. The searchers appear to have locked onto the locations of the four targets. The second figure shows the end of the scenario where the searchers have some idea, but not as good as the previous screenshot. This shows that searcher knowledge of target location went in cycles; the searchers had the targets pinned down, then the probability mass spread out, and eventually the searchers would pin down the targets again. This could also be explained by a high estimate of the probability of detection because it would eliminate too much mass from a location that was just searched when there should still be a significant probability mass at that location. If this estimate were lowered, it would take longer for the searchers to isolate the target location, but it would be

more accurate and unlikely to go through the cycle of target knowledge that was experienced in this trial.

Figure 11. Mid-Scenario Probability Map.

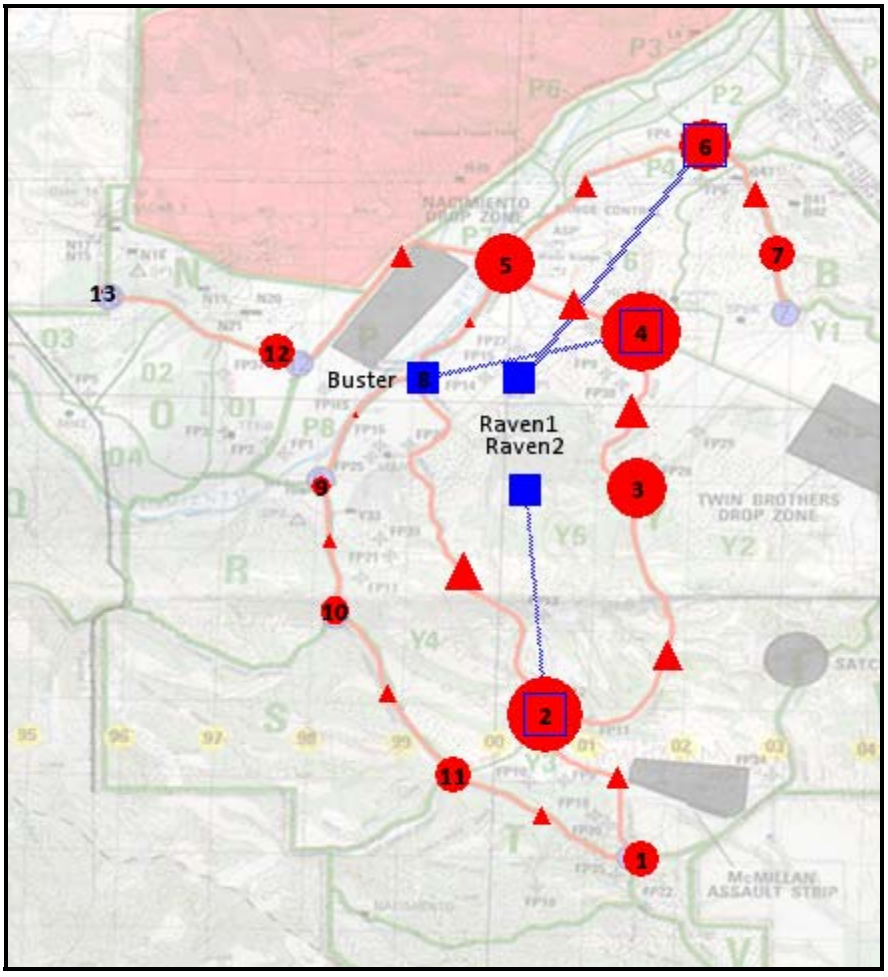
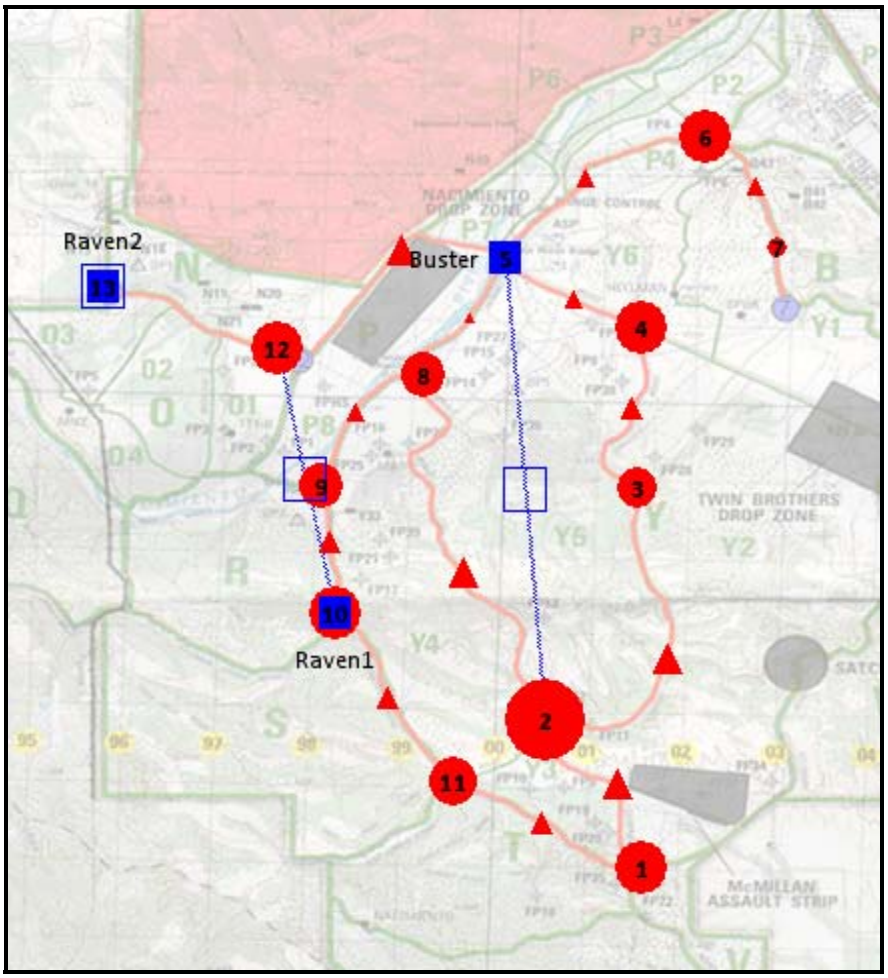


Figure 12. May Experiment Final Probability Map.



The second day's trial was markedly improved. The small problems we experienced in day 1 were fixed for day 2 and the long trial ran smoothly. During the trial, the UAVs operated without any mishaps. The disabled node was used for refueling purposes and worked according to plan. The results from day 2 were informative and the Excel interface made ASOM easier to understand, even for the people observing the experiment. After implementing the target thresholds, the searchers were able to concentrate their efforts on finding targets whose location probabilities were

spread out. The behavior of the searchers when they did not concentrate on searching nodes with recently found targets resulted in a noticeable improvement of situational awareness when compared to the greedier ASOM. Even after these updates, there are still a few recommendations for future work on ASOM.

THIS PAGE INTENTIONALLY LEFT BLANK

V. FINAL THOUGHTS

A. CONCLUSIONS

We have created a decision aid that recommends efficient search plans for multiple UAVs searching for multiple moving targets, possibly of different types. This decision aid demands few assumptions concerning the desired search scenario. ASOM is general enough to support many military or civilian search situations. It can be used to search for terrorists moving between safe-houses and friendly pilots who have been shot down in a wooded area. On the civilian side, it could be used for search and rescue missions after natural disasters or to search for lost hikers in the mountains. ASOM can also incorporate stationary searchers or targets and can even keep track of different types of targets. The decision aid is capable of being altered for a greedy search to keep track of targets once they are found, or to go after other targets that have not been found in a while, or at all.

Today, UAVs are increasingly used in combat situations. Their importance in future warfare will continue to grow and they are likely to become more important in many different civilian applications. Creating efficient search plans for these UAVs is the problem we chose to solve, but there are many other topics involving efficient UAV routing. There is a necessity for work such as that seen in this thesis and the importance of such work guarantees many different avenues for future research in this area.

B. FUTURE WORK

Currently, there are several aspects of ASOM that could be improved. Firstly, we did not take the wind speed and direction into account when determining flight times for UAVs to reach destination nodes. This update would involve creating a dynamic set of distance matrices that vary with wind speed and direction. This will make the calculations of arrival and search times far more accurate than the constant distances that we used in the calculations. While the wind factor is a relatively simple change to the model, it will dramatically increase the accuracy based on the amount of work required.

The second change would be to do some more calculations and experiments to get better estimates on the probability of detection for different UAVs. The values we used were estimated on past experience, but we believe them to be too high of an estimate. If more research was completed and better estimates found, again the accuracy of the model would be increased with a relatively small amount of work required, albeit somewhat time-consuming.

The third change would require a bit more programming experience, but in the end, could create the most accurate decision aid. This change would be to try and do more than the two-step look-ahead problem. Not to look further into the future, but to create an expected future reward based on the current state after the two-step look-ahead. This would be a way of estimating any further look-ahead based on the state as there are diminishing returns on looking further into the future and the computation time increases rapidly. This expected reward on future searches based on the state

is a good way to avoid the problem of computational complexity, yet get a more accurate solution.

A fourth possible change would be to try and incorporate target dependence into the model. Currently, the model assumes independent movement of the targets. This assumption makes computing the marginals based on movement from the Markov process easier than if the targets' movements were dependent on one another. Getting rid of this assumption would be a somewhat difficult task as that part of the updating phase would have to be reconstructed, but it would be a great way to extend our work on ASOM.

An extension to include different scenarios is to examine the possibility of tracking criminals after a robbery along city streets. In this scenario, searchers would first concentrate their search around the robbery location, but as time increases the graph of nodes and arcs would be forced to expand to represent the criminals getting away. There could even be an "escape node" to represent the criminals getting out of the area or exceeding the time the police are willing to search.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I: ADDITIONAL EXPRESSIONS FOR FORMULATION

Random Variables and Sets

ζ Random variable with a *uniform* $(0,1)$ distribution.

The following random data sets, DET_t, C_t, *and* $COMBO_{d,c,t}$ are used in the calculation of target detections. ASOM receives all of the detections as inputs during time t. ASOM must then determine the probability of each different possible scenario of detections occurring as explained in the model formulation. These calculations are handled by appropriate functions below, these are the random sets required for those calculations.

$DET_{b,t}$ Set of the number of detections of type b at time t, $b \in B$, $t \in T$, $DET_{b,t} = \{1, 2, ..., \sum_{i,j} D_{i,j,b,t}\}$, $d \in DET_{b,t}$.

$C_{b,t}$ Set of the number of different permutations of target detections of type b at time t, $b \in B$, $t \in T$, $C_{b,t} = \{1, 2, ..., |U|!/\left(|U| - |DET_{b,t}|\right)!\}$, $c \in C_{b,t}$.

$COMBO_{d,c,b,t}$ Matrix of the different permutations of target detections of type b at time t. Detection number d of permutation number c during time t, $c \in C_{b,t}$, $d \in DET_{b,t}$, $b \in B$, $t \in T$.

Model Formulation Functions

$$R_t(S_t, X_t) = \sum_{i,j,m} \left(AGG_{i,j,t} x_{i,j,m,t} z_{m,t} PD_{i,j,m} + AGG_{j,j,t} x_{i,j,m,t} PDET_{j,m}(\max(0, STEP - y_{m,t})) \right)$$

Reward for all searchers traveling between node i and j, $m \in M$, $i, j \in I$. The reward function is an important part of the model because it is what the model intends to optimize by changing the possible decision variables.

$$R^{ND}_{m,t}(S^{ND}_{m,t}, V_{m,t}) = \sum_{i,j} \left(AGG^{ND}_{i,j,m,t} x_{i,j,m,t} z_{m,t} PD_{i,j,m} + AGG^{ND}_{j,j,m,t} x_{i,j,m,t} PDET_{j,m}(\max(0, STEP - y_{m,t})) \right)$$

Reward for searcher m traveling between node i and j, $m \in M$, $i, j \in I$. This is the function used for the future reward where the state will depend on the previous moves of just one searcher.

$$PDET_{i,m}(\tau) = 1 - e^{\frac{-\tau SPEED_m SW_m}{\pi .0625}}$$

Probability of detection at node i by searcher m, dependent on amount of time searched, τ, $i \in I$, $m \in M$. This is the function used to determine probability of detection at a node, rather than on a road (handled earlier in the data section).

$$TOTAL_u(S_t, X_t) = \begin{cases} \sum_{i,j,m} \left(\begin{array}{l} MARG_{i,j,u,t} x_{i,j,m,t} z_{m,t} \left(1 - PD_{i,j,m}\right) + \\ MARG_{j,j,u,t} x_{i,j,m,t} \left(1 - PDET_{j,m}(\max(0, STEP - y_{m,t}))\right) \end{array} \right) & \max(MARG_{\bullet,\bullet,u,t}) < 1 \\ 1 & otherwise \end{cases} \quad \forall u$$

Sub-function of $NEGATIVE_{i,j,u,t}(S_t, X_t)$ and $NEGATIVE^{ND}_{i,j,u,m,t}(S_t, V_{m,t})$. It represents the normalizing factor, meaning it is the sum of all the posterior probabilities after a Bayesian update. If the variable array input is for a one searcher m (as with the input $V_{m,t}$), the summation over variable m is only over the single input value m.

$$NEGATIVE_{i,j,u,t}(S_t, X_t) = \begin{cases} \dfrac{MARG_{i,j,u,t}\left(1 - \prod_m \left(1 - PD_{i,j,m}\right)\right)}{TOTAL_u(S_t, X_t)} & i \neq j \\ \dfrac{MARG_{i,j,u,t}\left(1 - \prod_m \left(1 - PDET_{j,m}(\max(0, STEP - y_{m,t}))\right)\right)}{TOTAL_u(S_t, X_t)} & i = j \end{cases} \quad \forall i, j, u$$

Function to update probability maps for failed detection via Bayesian updating. Takes the posterior probabilities and normalizes by dividing by the sum of all posterior probabilities.

$$NEGATIVE^{ND}_{i,j,u,m,t}(S_t, V_{m,t}) = \begin{cases} \dfrac{MARG_{i,j,u,t}\left(1 - \prod_m \left(1 - PD_{i,j,m}\right)\right)}{TOTAL_u(S_t, V_{m,t})} & i \neq j \\ \dfrac{MARG_{i,j,u,t}\left(1 - \prod_m \left(1 - PDET_{j,m}(\max(0, STEP - y_{m,t}))\right)\right)}{TOTAL_u(S_t, V_{m,t})} & i = j \end{cases} \quad \forall i, j, u$$

Function to update probability maps for failed detection via Bayesian updating for look-ahead. Heuristic approach only takes into account the move of single searcher m.

$$TEMP1_{i,j,u,t}(S_t) = MARG_{i,j,u,t} MATRIX_{i,j} \qquad \forall i, j, u$$

Sub-function of $MARKOV_{i,j,u,t}(S_t)$ and $MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t})$. It represents the probability that a target at node *i* will remain at node *i* for the next timestep.

$$TEMP2a_{j,u,t}(S_t) = \sum_{i|i \neq j} \frac{MARG_{i,j,u,t}}{\max(TTS_{i,j,u}, 1)} \qquad \forall j, u$$

Sub-function of $TEMP2_{i,j,u,t}(S_t)$. It represents the additional probability each node will accumulate for the next timestep by the mass coming in from all adjacent roads.

$$TEMP2_{i,j,u,t}(S_t) = \begin{cases} TEMP2a_{j,u,t}(S_t) & i = j \\ 0 & i \neq j \end{cases} \qquad \forall i, j, u$$

Sub-function of $MARKOV_{i,j,u,t}(S_t)$ and $MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t})$. It extends the previous function, $TEMP2a_{j,u,t}(S_t)$, to account for the fact that only nodes, not arcs, have this property.

$$TEMP3_{i,j,u,t}(S_t) = \frac{(1 - TURN)\max(TTS_{i,j,u} - 1, 0) MARG_{i,j,u,t}}{\max(TTS_{i,j,u}, 0)} \qquad \forall i, j, u$$

Sub-function of $MARKOV_{i,j,u,t}(S_t)$ and $MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t})$. It represents the probability of target on arc (i, j) deciding to continue on that arc with $(1 - TURN)$ probability.

$$TEMP4_{i,j,u,t}(S_t) = \frac{TURN \max(TTS_{j,i,u} - 1, 0) MARG_{j,i,u,t}}{\max(TTS_{j,i,u}, 0)} \qquad \forall i, j, u$$

Sub-function of $MARKOV_{i,j,u,t}(S_t)$ and $MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t})$. It represents the probability of a target on arc (i, j) deciding to turn around with $TURN$ probability.

$$MARKOV_{i,j,u,t}(S_t) = \begin{cases} TEMP1_{i,j,u,t}(S_t) + TEMP2_{i,j,u,t}(S_t) & i = j \\ TEMP1_{i,j,u,t}(S_t) + TEMP3_{i,j,u,t}(S_t) + TEMP4_{i,j,u,t}(S_t) & i \neq j \end{cases} \qquad \forall i, j, u$$

Function to update probability maps for target movement based on Markov matrix. It incorporates all sub-functions to take into account for all probability mass leaving and coming into arc (i, j).

$$MARKOV^{ND}_{i,j,u,m,t}(S^{ND}_{m,t}) = \begin{cases} TEMP1_{i,j,u,t}(S^{ND}_{m,t}) + TEMP2_{i,j,u,t}(S^{ND}_{m,t}) & i = j \\ TEMP1_{i,j,u,t}(S^{ND}_{m,t}) + TEMP3_{i,j,u,t}(S^{ND}_{m,t}) + TEMP4_{i,j,u,t}(S^{ND}_{m,t}) & i \neq j \end{cases} \qquad \forall i, j, u, m$$

Function to update probability maps for target movement based on Markov matrix for the second step look-ahead. It incorporates all sub-functions to take into account for all probability mass leaving and coming into arc (i, j). The only difference between this function and the normal MARKOV function, is that this one is performed for each searcher m, and the current input of the state will only be updated for the moves of searcher m.

$$PR1_{i,j,d,c,t}(S_t, D_{\bullet,\bullet,b,t}) = \begin{cases} MARG_{i,j,COMBO_{d,c,b,t},t} & \sum_{k1=1}^{i}\sum_{k2=1}^{j-1} D_{k1,k2,b,t} < d \le \sum_{k1=1}^{i}\sum_{k2=1}^{j} D_{k1,k2,b,t} \\ 1 & otherwise \end{cases} \qquad \forall i, j, d, c$$

Sub-function of $PR2_{i,j,c,t}(S_t, D_{\bullet,\bullet,b,t})$. It calculates the probability of seeing a particular target of a particular combination c for detection d of that combination for each arc (i, j).

$$PR2_{i,j,c,t}(S_t, D_{\bullet,\bullet,b,t}) = \prod_{d} PR1_{i,j,d,c,t}(S_t, D_{\bullet,\bullet,b,t}) \qquad \forall i, j, c$$

Sub-function of $PR3_{c,t}(S_t, D_{\bullet,\bullet,b,t})$. It determines the total probability of seeing all detections of a particular combination for each arc (i, j).

$$PR3_{c,t}(S_t, D_{\bullet,\bullet,b,t}) = \prod_{i,j} PR2_{i,j,c,u,t}(S_t, D_{\bullet,\bullet,b,t}) \qquad \forall c$$

Sub-function of $PR4_{c,t}(S_t, D_{\bullet,\bullet,b,t})$. It determines the total probability (no normalization) of seeing each combination of target detections by multiplying over i and j.

$$PR4_{c,t}(S_t, D_{\bullet,\bullet,b,t}) = \frac{PR3_{i,j,c,u,t}(S_t, D_{\bullet,\bullet,b,t})}{\sum_{c1 \in C_t} PR3_{i,j,c1,u,t}(S_t, D_{\bullet,\bullet,b,t})} \qquad \forall c$$

Sub-function of $CHOICE_{d,c,t}(S_t, D_{\bullet,\bullet,b,t})$. It normalizes the calculation of $PR3_{c,t}(S_t, D_{\bullet,\bullet,b,t})$.

$$CHOICE_{d,c,t}(S_t, D_{\bullet,\bullet,b,t}) = \begin{cases} COMBO_{d,c,t} & \sum_{c1=1}^{c-1} PR3_{c,u,t}(S_t, D_{\bullet,\bullet,b,t}) < \zeta < \sum_{c1=1}^{c} PR3_{c1,u,t}(S_t, D_{\bullet,\bullet,b,t}) \\ 0 & otherwise \end{cases} \quad \forall d, c$$

Sub-function of $CHOICE2_{d,t}(S_t, D_{\bullet,\bullet,b,t})$, ζ denotes a random number drawn from a uniform(0,1) distribution. It determines the actual scenario of target detections that occurred according to the model based on this random draw and the probabilities of each scenario occurring by setting all other combination values to 0.

$$CHOICE2_{d,t}(S_t, D_{\bullet,\bullet,b,t}) = \sum_{c} CHOICE_{d,c,t}(S_t, D_{\bullet,\bullet,b,t}) \quad \forall d$$

Sub-function of $POS1_{i,j,d,u,t}(S_t, D_{\bullet,\bullet,b,t})$. It gets rid of all other combinations except the values of the one that actually occurred.

$$POS1_{i,j,d,u,t}(S_t, D_{\bullet,\bullet,b,t}) = \begin{cases} 1 & \left(\sum_{k1=1}^{i} \sum_{k2=1}^{j-1} D_{k1,k2,b,t} < d \le \sum_{k1=1}^{i} \sum_{k2=1}^{j} D_{k1,k2,b,t} \right) \&\& \\ & \left(u = CHOICE2_{d,t}(S_t, D_{\bullet,\bullet,b,t}) \right) \\ 0 & otherwise \end{cases} \quad \forall i, j, d, u$$

Sub-function of $POS2_{i,j,u,t}(S_t, D_{\bullet,\bullet,b,t})$. It stores the value 1 for all locations that a detection occurred at arc (i, j) for target u and detection d and zero otherwise.

$$POS2_{i,j,u,t}(S_t, D_{\bullet,\bullet,b,t}) = \sum_{d} POS1_{i,j,d,u,t}(S_t, D_{\bullet,\bullet,b,t}) \quad \forall i, j, u$$

Sub-function of $POSTYPE1_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t})$. It sums over the detection number variable so we have a 1 if a detection occurred on arc (i, j) for target u.

$$POSTYPE1_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t}) = \begin{cases} MARG_{i,j,u,t} & \sum_{i1,j1 \in I} POS2_{i1,j1,u,t}(S_t, D_{\bullet,\bullet,b,t}) < 1 \\ POS2_{i,j,u,t}(S_t, D_{\bullet,\bullet,b,t}) & otherwise \end{cases} \quad \forall i, j, u, b$$

Sub-function of $POSTYPE2_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t})$. It sets the value equal to the marginal value if no detection occurred and 1 if a detection did occur (to spike the probability) for each target type *b*.

$$POSTYPE2_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t}) = \begin{cases} POSTYPE1_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t}) & TYPE_u = b \\ 0 & otherwise \end{cases} \quad \forall i, j, u, b$$

Sub-function of $POSITIVE_{i,j,u,t}(S_t, D_{\bullet,\bullet,\bullet,t})$. It sets the target marginals to the correct values only if the current target being looked at, *u*, matches the current type, *b*.

$$POSITIVE_{i,j,u,t}(S_t, D_{\bullet,\bullet,\bullet,t}) = \sum_b POSTYPE2_{i,j,u,b,t}(S_t, D_{\bullet,\bullet,\bullet,t}) \quad \forall i, j, u$$

Function to update probability maps for positive detection via Bayesian updating. It sums over the probabilities for different types of targets.

APPENDIX II: MATLAB FUNCTION DESCRIPTIONS

A. STEP.M FUNCTION

This function is the main workhorse that runs the algorithm. It does all calculations, either inside the function, or calling other functions to do the work for it. It first updates the target marginals by running the positive Bayesian updates (detections) for different target types (PositiveBayesianPermutations.m). The function then makes all essential updates to the probability of detection at each arc (i, j), including the nodes and connecting arcs for any stationary searchers using the locations of each searcher. After these steps, the function updates the target marginals for negative Bayesian updates (NegativeBayesian.m), the traditional application of Bayes' theorem. Next, the function updates for target movement from the Markov process (MarginalsMovement.m) to account for the fact that targets could have moved during the current timestep. Finally, the function determines which moves to recommend for the next timestep with the current state and detection matrix (MultiSearcherMove.m).

B. INITIALIZEMARGINALS.M FUNCTION

This function only serves a purpose for the actual experiment. It is a way to initialize the target marginals before an experiment begins. It takes as an input, the number of targets that are going to be involved in the experiment and returns the resulting initial target marginals. For our experiments, we assumed a target was ten

times more likely to start at a node than on a road, but this value is completely dependent on the conditions of the scenario.

The function calculates these initial conditions by creating an integer count on each arc (i, j) to represent how likely it is to start there. Giving a value of 10 to each node, 1 to each road, and zero at every other (i, j). It then divides by the sum total of the entire matrix to convert these counts into probabilities. Finally, it sets these probabilities for all targets.

C. AREASEARCH.M FUNCTION

This is a simple function that determines the probability of detection at a node given the time spent searching at the node as well as the speed and sweepwidth of the searcher. It does this by using the random search formula assuming a circular search area of radius one-quarter mile around the node. We assumed a random search to calculate a lower bound on the actual probability of detection. This function is used in the SearcherMove.m function to help determine how much probability mass would be collected by a certain move.

D. SEARCHERMOVE.M FUNCTION

This function takes in the state and characteristics of one particular searcher as well as a list of nodes not available for this searcher at this time. It returns the searchers best first and second moves (second move refers to the move in the next timestep, which will be reoptimized based on the actual state during the next timestep), as well

as how much probability mass these moves collect and whether this sequence of moves takes both timesteps, thus constraining the options for the next timestep's move.

The function works by looping through all nodes and checks which ones the searchers are able to transit or conduct a road search to during the next timestep. It accomplishes this by using two nested "for" loops. It then updates the target marginals with a negative Bayesian update function, thus inherently assuming no detections were made in this timestep in order to get a more accurate estimate of the state for the next timestep (this assumption is not made during the reoptimization of the future move, it is merely made now for a more accurate representation of the future state). The function then uses two more nested "for" loops inside of the other two to calculate every sequence of two moves (still including the option of either transiting or searching the road) and determines the reward of doing such a sequence of moves. If the sequence of moves the function is currently examining is better than any previous sequence, it stores these moves as the current best. It then repeats this process until all moves have been checked.

E. MULTISEARCHERMOVE.M FUNCTION

This function takes in the number of searchers and their characteristics as well as the state at the current time. It returns the recommended move for the current timestep for each searcher and whether or not that searcher will be blocked (constrained to continue along that search) for the next timestep.

The function accomplishes this by repeatedly calling the SearcherMove.m function with different restrictions for each unconstrained searcher (searchers can be constrained if their previous move limits their next move, i.e., they are still en route to their previous destination, or if they are currently inactive, i.e., out of fuel or down). The function first limits constrained searchers to their appropriate moves and then updates the restricted movement list to incorporate these moves. It will get the optimal move for each searcher by running the SearcherMove.m function and storing these optimal moves. If there are no conflictions, these are the optimal moves for the searchers; if there are conflictions, the function will then update the list of unavailable moves for each searcher and determine the best scenario possible using these conflicting searchers. It will repeat this process until there are no conflictions among the searchers and this will be the recommended movements for the next timestep. This iterative process of eliminating possible moves and recalculating optimal moves for each searcher can save orders of magnitude in runtime over the total enumeration method for all searchers combined which tries many moves that are nowhere near optimal strategies. Even in the TTLP, total enumeration for a real-time experiment can take too long, thus this iterative optimal move process is an extremely important process of the ASOM algorithm.

F. POSITIVEBAYESIANPERM.M FUNCTION

This function takes in the current target marginals and a matrix of all the detections. It returns the resulting target marginals after updating for the positive detections

in the current timestep. It is only appropriate to use this function when all targets are of the same type, the more general type of this function and the one that is used in practice is PositiveBayesianPermutations.m.

This function works by creating a matrix of all different (unordered) combinations of targets that could have been seen during the timestep using the nchoosek.m MATLAB library function. Next, for each different combination (each row of the previously created matrix) it creates all different permutations (ordered) of that combination using the perms.m MATLAB library function. It combines all of these different permutations into one big matrix of all possible permutations for the target detections of the current timestep. It is important to notice that these permutations represent all of the different possible scenarios of target detections. The function then determines the probability of each of these scenarios occurring by multiplying together the target marginals of each detected target at the location it was supposedly detected then normalizing by dividing each probability by the sum total of all probabilities. After determining and normalizing the probabilities, the function decides which scenario actually occurred (according to the model/algorithm’s viewpoint) based on a random number draw. Now that the algorithm has the scenario that occurred picked out, it updates the target marginals for all targets that were detected to be one at the arc they were detected and zero everywhere else, thus spiking the probability of those targets.

G. POSITIVEBAYESIANPERMUTATIONS.M FUNCTION

This function takes in the current target marginals and an array containing the information of each target type, as well as a list of all detection locations and the type of detection made at each location. It returns the resulting target marginals after all positive Bayesian updates have been made.

The function works by creating new temporary target marginal matrices with an extra index representing all possible types of targets. This will create many blank (by blank, we mean no nonzero entries) levels of the target marginals of each type, as there will only be nonzero entries if the target type of the marginals index matches the actual type of the target. In a similar manner, the function also creates a temporary detection matrix with an extra index to indicate detections of a certain type of target. Next, the function calls the PositiveBayesianPerm.m function for each type separately, meaning where the PositiveBayesianPerm.m function is expecting the input of the target marginals and a matrix of detections, we only give it one level of the temporary target marginals and temporary detection matrix by holding the type index fixed at its current value and looping through all possibilities. This updates the temporary target marginals for each type separately, but since all values were zero except for targets whose type matched the current type index, we simply have to sum over the type index to return the final value of the actual target marginals updated for positive detections.

H. NEGATIVEBAYESIAN.M FUNCTION

The NegativeBayesian.m function is the Bayesian update for nondetection function. This is the traditional use of Bayesian updating as described in the introduction. It takes all values of target marginals where there was no detection and updates them for the failed detection. The function returns the updated values of the target marginals.

The function accomplishes this by looking at every value of the target marginals that is less than 1, meaning if there was a detection there (thus giving a probability spike equal to 1), do not apply negative Bayesian updating. If the value of the target marginal is less than 1, the function updates this probability to its previous value multiplied by the probability of failed detection (1 - probability of detection). After updating the probability of each target marginal, the function normalizes each value by dividing it by the sum total of the new probabilities. The result is the new target marginals updated for failed detections.

I. MARGINALSMOVEMENT.M FUNCTION

The MarginalsMovement.m function takes in the current target marginals as well as the speed of each of the targets and returns the updated values of the target marginals after incorporating possible movement for the current timestep based on the Markov movement matrix.

The function accomplishes this by looping through each target and another loop through each arc (i, j) for that target. First, it updates every arc to the new value based on movement out of it for the next timestep by multiplying

by the movement matrix directly. Next, it updates the values of nodes that have some probability moving into them from adjacent roads. After that, it multiplies the values on roads by $(1-TURN)$ probability to lessen the values on arcs where the target could possibly turn around. Finally, on every arc where it lowered the probability to account for targets turning around, it raises the probability on the reverse arc by the corresponding amount.

J. MOVEMENT.M FUNCTION

The Movement.m is one of two functions to help model the target movement for experimentation. It is not actually used in the step function, nor during the actual experiment, but rather to aid in the generation of random routes for targets to travel during experimentation. It is called in the TargetMovement.m function to return the next move of a target that needs a new destination. It takes in the old position of the target and the Markov movement matrix. It returns the new destination node of that target.

This function works by looking at the Markov movement matrix in the row of the starting position of the target (which will sum to 1, by definition) and making a random draw from a uniform(0,1) distribution. With this random number, the function returns the column of the number whose cumulative probability matches with the random number drawn.

K. TARGETMOVEMENT.M FUNCTION

The second of two functions made to model target movement for experimentation. It takes in the amount of time the targets will move around, the number of targets, a

speed array containing the speed of each target, and the starting positions of the targets. It returns the final positions of the targets after it has moved for the amount of time input. The output matrix has one row for each target and three columns with the first two representing the start and finish nodes of the current arc the target is on (if start and finish nodes are equal, the target is stationary at that node), and the third being how many timesteps the target has remaining on that arc before completing it. If the user would like to see every movement in the sequence, just repeatedly run the function with end time equal to one timestep and update the start positions with the output positions from the previous step.

This function works by entering a “while” loop until the simulation time reaches the end time input. It then loops through each target to update their positions one at a time. If the current target is stationary at a node, it calls the movement function to get a new destination node (which could be to remain at the same node for another timestep), otherwise the target remains on the road it was previously located. It then makes a draw from a uniform(0,1) distribution, if this random draw is less than the turn probability, the function reverses the arc and number of moves remaining to complete that arc, otherwise the function only updates the number of moves remaining until completion of its current arc. Finally, the function stores all of the new information in the output matrix and increments time for the next timestep.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] F. Bourgault, T. Furukawa, and H.F. Durrant-Whyte. Coordinated decentralized search for a lost target in a Bayesian world. *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 48-53, 2003.

[2] X. Cao and X. Guo. Partially observable Markov decision processes with reward information. *Proceedings IEEE Conference on Decision and Control*, 4393-4398, 2004.

[3] T.H. Chung and J.W. Burdick. A decision-making framework for control strategies in probabilistic search. *Proceedings IEEE International Conference on Robotics and Automation*, 2007.

[4] R.F. Dell, J.N. Eagle, G.H.A. Martins, and A.G. Santos. Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43:463-480, 1996.

[5] J.N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32:1107-1115, 1984.

[6] J.N. Eagle and J.R. Yee. An approximate solution technique for the constrained search path moving target search problem. Naval Postgraduate School Technical Report NPS-55-85-015, 1985.

[7] J.N. Eagle and J.R. Yee. An optimal branch and bound procedure for the constrained path, moving target search problem. *Operations Research*, 38:110-114, 1990.

[8] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand. Cooperative control for multiple autonomous UAV's searching for targets. *Proceedings IEEE Conference on Decision and Control*, 2823-2828, 2002.

[9] M. Flint, E. Fernandez, and M. Polycarpou. Efficient Bayesian methods for updating and storing uncertain search information for UAV's. *Proceedings IEEE Conference on Decision and Control*, 1093-1098, 2004.

[10] T. Furukawa, F. Bourgault, B. Lavis, and H.F. Durrant-Whyte. Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. *Proceedings IEEE International Conference on Robotics and Automation*, 2521-2526, 2006.

[11] Y. Jin, A.A. Minai, M.M. Polycarpou. Cooperative real-time search and task allocation in UAV teams. *Proceedings IEEE Conference on Decision and Control*, 7-12, 2003.

[12] M. Kress and J.O. Royset. Aerial search optimization model (ASOM) for UAVs in special operations. *Military Operations Research*, 13(1):23-33, 2008.

[13] H. Lau, S. Huang, G. Dissanayake. Discounted MEAN bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operations Research*, 190(2):383-397, 2008.

[14] W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley & Sons, Inc., Hoboken, NJ, 2007.

[15] J.R. Riehl, G.E. Collins, J.P. Hespanha. Cooperative graph-based model predictive search. *Proceedings IEEE Conference on Decision and Control*, 2298-3004, 2007.

[16] A. Ryan, J. Tisdale, M. Godwin, D. Coatta, D. Nguyen, S. Spry, R. Sengupta, and J.K. Hedrick. Decentralized control of unmanned aerial vehicle collaborative sensing missions. *Proceedings American Control Conference*, 4672-4677, 2007.

[17] H. Sato and J.O. Royset. Path optimization for the resource-constrained searcher. In review.

[18] T.J. Stewart. Search for a moving target when searcher motion is restricted. *Computational and Operations Research*, 6:129-140, 1979.

[19] T.J. Stewart. Experience with a branch-and-bound algorithm for constrained searcher motion. In *Search Theory and Applications*, K.B. Haley and L.D. Stone (eds.). Plenum Press, New York, 1980.

[20] D.H. Wagner, W.M. Mylander, and T.J. Sanders. *Naval Operations Analysis*, Naval Institute Press, Annapolis, MD, 187-212, 1999.

[21] A.R. Washburn. Branch and bound method for a search problem. *Naval Research Logistics*, 45:243-257, 1998.

[22] A.R. Washburn. *Search and Detection 2nd edition*, ORSA Books, Arlington, VA, 1982.

[23] E. Wong, F. Bourgault, and T. Furukawa. Multi-vehicle Bayesian search for multiple lost targets. *Proceedings IEEE International Conference on Robotics and Automation*, 3169-3174, 2005.

[24] W.I. Zangwill. The convex simplex method. *Management Science*, 14(3):221-238, 1967.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Assistant Professor Johannes O. Royset
   Naval Postgraduate School
   Monterey, California

4. Professor Moshe Kress
   Naval Postgraduate School
   Monterey, California

5. Assistant Research Professor Timothy H. Chung
   Naval Postgraduate School
   Monterey, California

6. Professor Emeritus David Netzer
   Naval Postgraduate School
   Monterey, California